Tartalomjegyzék

Ismétlés az elõadásról

- Egyszerû adattípusok: None, int, long, float, complex
- Összetett adattípusok: str, list, tuple, set, dict
- Lista mûveletek: [], [:], len(), range()
- Tuple elemei változtathatatlanok
- Halmaz: .remove(), in
- Szótár: [], in, .keys(), .values()
- type függvény
- Feltételes utasítás: if, elif, else
- While ciklus
- For ciklus

- 2. el?adás

Lista és tuple alapok

1. Adj meg egy legalább 5 elem?, egész számokat tartalmazó listát, és rendeld az *L* változóhoz!
2. Írasd ki a lista második elemét!
3. Írasd ki a lista második, harmadik, és negyedik eleméb?l álló részlistát (használd a kett?spontot a szögletes zárójelen belül)!
4. Írasd ki a lista els? 3 elemét!
5. Írasd ki a lista utolsó elemét (negatív index)!
6. F?zz a lista végére egy új elemet, értéke legyen ugyanaz, mint az els? elem! (*append()*)
7. Keresd meg hogy egy elem hányadik indexen szerepel a listádban! (*index()*)
8. Számold meg, hányszor szerepel az els? elem a listában! (*count()*)
9. Mennyi a listádban szerepl? számok összege? (*sum()*)
10. Rendezd a listádat növekv? sorrendbe! (*sort()*)
11. F?zd össze az *L* listát az [1,2,3] listával! (használd a + operátort)
12. Készíts listát (*A* néven) az "http://wiki.math.bme.huabrakadabra"http://wiki.math.bme.hu stringb?l! (*list()*)
13. Készíts stringet az *A* listából! (*str()*)
14. Készíts tuple-t *T* néven az *A* listából, majd írd ki az utolsó elemét!
15. Változtasd meg a *T* els? elemét!

Vezérlési szerkezetek (if, for, while)

- Egészítsd ki a függvényt, hogy ha az elsõ paraméter kisebb mint a második, akkor azt a karakterláncot adja vissza, hogy "http://wiki.math.bme.hukisebb"http://wiki.math.bme.hu, ha nagyobb, akkor azt, hogy "http://wiki.math.bme.hunagyobb"http://wiki.math.bme.hu és ha egyenlõ akkor "http://wiki.math.bme.huegyenlo"http://wiki.math.bme.hu-t!

```
<!> hasonlit(a, b):
    if a < b:
        <!> "http://wiki.math.bme.hukisebb"http://wiki.math.bme.hu
    elif a > b:
        <!> "http://wiki.math.bme.hunagyobb"http://wiki.math.bme.hu
    <!>:
        <!> "http://wiki.math.bme.huegyenlo"http://wiki.math.bme.hu
```

- Definiálj egy Sage függvényt *elojel* néven, amelynek egy bemenete van (*a*), és a "http://wiki.math.bme.hupozitiv"http://wiki.math.bme.hu karakterláncot (vagyis stringet) írja ki ha a kapott paraméter pozitív, "http://wiki.math.bme.hunegativ"http://wiki.math.bme.hu-at ad vissza ha a szám negatív, és "http://wiki.math.bme.hunulla"http://wiki.math.bme.hu-ta ad ha nulla volt a paraméter értéke.
- Írj egy Sage függvényt *primek_szama* néven, amely bemenetként kap egy számot (*n*) és visszaadja az n-nél kisebb prímek darabszámát! (Segítség: a range függvényt, és for ciklust használj)
- Írj egy függvényt *elso_primek* néven, amely bemenetként kap egy számot (*n*) és kiírja az els? *n* prímszámot! (Segítség: itt egyszerûbb ha while ciklust használsz)

Listák kezelése

- Egészítsd ki a kódot, hogy a függvény az elsõ paraméterként kapott listának csak minden *n*-edik elemeibõl álló listát adja vissza, ahol *n* a második paraméter!

```
def kivalaszt(l, n):
    l2 = []
    hossz = <!>
    <!> e in range(n, hossz, n):
        l2.<!>(l[e])
    return <!>
```

- Ezek után lássuk be, hogy a fenti függvény azonos kimenetet ad mint a következõ (ebben a feladatban csak végig kell gondolni a következõ függvényt):

```
def kivalaszt2(l, n):
    l2 = []
    i = n
    while i < len(l):
        l2.append(l[i])
        i += n            # ez ekvivalens a következõvel i = i + n
    return l2
```

- Írj függvényt *sokszorozo* néven, amely bemenetként kap egy számot (*n*) és még egy paramétert, *a*-t (ennek a típusa bármi lehet). A függvény adjon vissza egy listát, amiben *n*-szer szerepel az *a* értéke. Például: *sokszorozo(3, "http://wiki.math.bme.hubla"http://wiki.math.bme.hu)* kimenete ["http://wiki.math.bme.hubla"http://wiki.math.bme.hu, "http://wiki.math.bme.hubla"http://wiki.math.bme.hu, "http://wiki.math.bme.hubla"http://wiki.math.bme.hu] legyen.

- Írj függvényt *reszlista* néven, amely bemenetként kap egy listát, és a függvény kimenete az a részlistája legyen az eredetinek, amely a két els? *0* közötti elemekb?l áll. Ha nincs a bemeneti listában legalább 2 nulla, akkor legyen a kimenet az üres lista.
  Például *reszlista(1, 4, 0, 3, 5, 6, 3, 0, 23, 5, 0, 1, 0, 4)* eredménye legyen *[3, 5, 6, 3]*
- Írj Sage függvényt amely megfordít egy bemenetként kapott listát!

Szótárak kezelése

- Legyen egy *gyumolcs_arak* nev? szótárunk, a következ? kulcs-érték párokkal:

  'alma': 150
  'szilva': 190
  'ananász': 450
  'banán': 300

- És legyen egy másik, *vasarlas* nev? szótár, amely azt tárolja, mib?l mennyit vettünk:

  'banán': 0.6
  'alma': 1.5
  'ananász': 2

- Írj Sage függvényt (legyen a neve *ar_szamolo*), amely megkapja a fenti két szótárat (els? paramétere legyen az árakat tartalmazó), és kiszámolja, hogy mennyit kell fizetnünk a gyümölcsökért!

Összetettebb feladatok

Tökéletes számok

- Írj függvényt, amely egy *a* számról eldönti, hogy az tökéletes szám-e (megegyezik a nála kisebb osztóinak az összegével, pl: 6, 28).
- Írj függvényt, amely 1-t?l *n*-ig összegy?jti és egy listában visszaadja a talált tökéletes számokat!

A 3n+1 probléma

A híres 3x+1 probléma (Collatz-sejtés) : végy egy számot, ha páratlan, szorozd meg 3-mal és adj hozzá 1-et, ha páros, oszd el 2-vel. Az az állítás, hogy így bármilyen pozitív egész számból indulva el?bb-utóbb eljutunk 1-ig.

Írj Sage függvényt, amely *x*-et kap bemenetként, és sorban kiírja a lépéseket 1-ig!